Basu 1-1

REMARKS

Claims 4 and 30 were rejected under 35 USC 112, second paragraph. The claims are amended herein to overcome the rejection.

Claims 30-31 were rejected under 35 USC 102 as being anticipated by Eliott, US Patent 6,468,160. Applicants respectfully traverse.

The Examiner asserts that Eliott describes a storage medium that sotres a control routine for use by a system to assure security of the system, and further asserts that the this storage device includes instructions for booting a standalone host with an authenticated operating device, citing col. 10, lines 26-56, and quoting particularly lines 47-49. The cited passage and the text pertaining to it teach that game console 52 (see FIG. 2) includes a main processor, and a peripheral interface 138. The peripheral interface has a small amount of initial program code that is executed by processor 100 at time of startup. This allows processor 100 to execute

> game program instruction 108 within storage device 53. The initial game program instruction 108 may, in turn, control main processor 100 to initialize the drivers and controllers it needs to access main memory 100. Co. 10, lines 52-56.

The Examiner has not explicitly stated what element corresponds to the "storage medium" of claim 30, nor what is the "standalone host" (which amended claim 30 now correctly identifies as the system identified in the claim's preamble). However, because the Examiner refers to the instructions in the ROM 138, it follows that the Examiner is asserting that ROM 138 (or some other device that includes ROM 138) corresponds to the claimed storage medium. Since ROM 138 is within console 52, it follows that the Examiner asserts that ROM 138, or the console 52 and ROM 138 combination, corresponds to the storage device of claim 30.

According to amended claim 30, the instructions boot the system identified in the claim's preamble, since the instructions of ROM 138 are executed in main processor 100, it follows that the Examiner effectively asserts that processor 100 (or some device that contains processor) is "the system."

The second clause of claim 20 specifies that the instructions in ROM 138 verify an operating system of the system. To maintain the Examiner's correspondence, it must be that the instructions in ROM 138 (or some device that includes ROM 138) verify an

operating system of processor 100 (or some device that includes processor 100). The Examiner points to col. 26, lines 10-16, which state:

> Resident in boot ROM 182 is a set of instructions which permit the remainder of the expansion device operating system to be accessed. After authentication has occurred, the operating system stored in hard drive 206 is accessed. The operating system of the video game system 50 is likewise authenticated so that the presence of authentic code in both the video game system and expansion device is verified.

The passage quoted above is ambiguous in that it does not tell what "authentication" is referred to in the line 3 phrase "after authentication has occurred," and it is not clear where is the "operating system of the video game system 50." However, it is perfectly clear that the passage teaches about ROM 182, which, of course, is not ROM 138. ROM 182 is a ROM within game expansion device 95 (see FIG. 4), whereas ROM 138 is located in console 52. Console 52 is coupled to expansion device 95 to form game system 50.

It is noted that the only reference to a location of an operating that is found in columns preceding the above-quoted language refers to an operating system on mass storage device 174 WITHIN the expansion device.

Based on the above, applicants respectfully submit that correspondence between claim 30 and the Eliott reference fails because the storage device that allegedly corresponds to the claimed storage device, that is ROM 138, and which boots the system (allegedly booting processor 100, which corresponds to the claim's "system") does not do any verifying of an operating system. Rather, by the Examiner's own assertion, the device that does the verifying is ROM 182, which is within expansion device 95 and which verifies the operating system that is also stored in expansion device 95.

As for the next clause of claim 30, which specifies that the instructions of the storage device transfer control of the system from the operating system of the storage device (herein, AOS – which stands for "Authenticating Operating System") to the operating system of the system (herein, VOS – which stands for "Verified Operating System"), the Examiner cites FIG. 17 and asserts that the FIG. describes something "wherein once the verification procedure has been completed, control shifts from the kernel to the operating system allowing the system to run the desired applications."

Basu 1-1

FIG. 17, however, shows nothing that is referred to as an "operating system," "verification procedure," or "kernel," so the text relative to FIGS. 17a and 17B was perused (col. 37, line 8 – col. 41, line 9). This text mentions verification of game ownership (co. 39, line 31), and verification of permissions to use a game (col. 39, line 54). There is no mention of verifying an operating system, and there is no mention of a kernel. The only mention of the operating system is found in col. 40, line 10, where it is taught that the security system employs API calls to expansion device 95, and that the API calls "may be built into the operating system for video game 50." Which element within video game 50 holds the operating system is not disclosed. It can only be assumed that the reference is to the operating system within mass storage device 174 that is within the expansion device.

What is quite clear, though, is that the Eliott reference does not do a verification of an operating system of a given system (VOS) by use of instructions and an operating system that reside on a particular storage medium (AOS), and does not execute a transfer of control from the AOS to the VOS.

Therefore, it is respectfully submitted that claim 30 is not anticipated by the Eliott reference.

As for claim 31, the Examiner points to FIG. 17 and, in particular, to elements 514, 518, 520, and 528, which, according to the Examiner, teaches "a verification procedure [that] uses a hash value to verify the operating system." Applicants respectfully disagree. FIG. 17 shows that various calls are made to the expansion device, and the expansion device returns a hash value. That is not a verification of an operating system. The elements specifically cited by the Examiner combine to check the hash results of the various calls. This, too, is not a verification of an operating system. Furthermore, claim 31 clearly specifies that it is the executable modules of the operating system that are verified though hashing. In contradistinction, the application modules that are verified by Eliott in FIG. 17 are not operating system executable modules. To illustrae, building a game request packet is NOT an operating system executable module. Consequently, applicants respectfully submit that claim 31 is not anticipated by the Eliott reference.

Basu 1-1

It is noted that both claims 30 and 31 are amended herein. However these amendments are included not in an effort to overcome the prior art – as the arguments above demonstrate – but, rather, to correct ambiguities, and to present the claims in clearer language.

Claims 1-16, 12-15, and 25 were rejected under 35 USC 103 as being unpatentable over Hanif et al, US Patent 6,141,667 in view of Johansson et al, US Paent 6,466,559. Applicants respectfully traverse.

With reference to claim 1, the Examiner asserts that the plurality of stacks that claim 1 specifies is taught by Hanif et al in col. 1, lines 58-67, and in col. 4, lines 24-29; that the service director that claim 1 specifies is taught by Hanif et al in col. 4, line 30 through col. 6, line 22, supported by the Johansson et al teachings at col. 15, line 49 through col. 16, line 13, with each one of the Johansson resources being responsive to requests of a different type. Applicants respectfully traverse.

The Hanif et al reference teaches a file server that is implemented with a multi-threaded process. It contemplates serving requests from clients that access the server from network. A client that wishes to communicate with the server accesses a Session Listening (SLS) socket on the server. Through the SLS socket a Server Session Socket (SSS) is established. For each established session, a session object (ASP, which stands for AppleTalk Session Protocol) is established. Thereafter, requests can be accepted from the server, and each requests is embodied in an AFP object (AFP stands for AppleTalk File Protocol).

In the past, according to Hanif et al, a request to a file server was satisfied immediately, unless the file server was busy with a previously submitted request. A queue held the requests that arrived but did not get immediately satisfied. To improve the response time when there is a plurality of tasks to be satisfied, some systems use multi-threaded processing, and each ASP is assigned to its own thread. See col. 1, lines 59-67. According to col. 5, lines 12-14, "In a multi-threaded process, there are multiple schedulable threads of control that share both address space and resources."

Basu 1-1

In those unidentified prior art systems that are mentioned in col. 1, the threads are embodied in stacks (col. 1, line 64). However, it should be noted that while a thread can be implemented with a stack, it does not have to be.

Hanif et al suggest a modification with a structure like the one shown in FIG. 6, where a number of threads are maintained, but that number is significantly lower than the total number of active sessions. That is the main thrust of the advance over the prior art that is asserted by Hanif et al.

The Hanif et al structure comprises a number of threads, which are shown in FIG. 6 as a single block, and not described as a stack. The structure also comprises a queue (Q1) of all open sessions, a queue (Q2) of all active sessions, and a separate queue (Q3) for each of the active sessions. Each of these queues contains the AFP entities that belong to the active session. The open sessions are assigned to the threads by cycling through the threads as the sessions are established and assigning the sessions on a first come, first serve basis.

In operation, this arrangement cycles through the threads, and as each thread comes up for execution, an active session from the top of queue Q2 that had been assigned to the thread is considered, for example ASP-x, and its associated FIFO queue Q3 is accessed to pop the top most (oldest) AFP. That AFP is processed, the handled session ASP-x identifier is placed at the bottom of the Q2 queue, the next thread comes up for execution, and the process repeats.

Although Hanif et al refer to Q2 as a queue, it appears to not be a classic queue, where items are not taken in a prescribed order (FIFO, or LIFO) that is based strictly on the items' position in the queue. In Q2, rather, there is an ordering, but it is a combination of the position in the queue AND the threads to which the items are assigned that identifies the item that is selected out of the queue. For example, if items are found in Q2 in the order ASP-1, ASP-3, ASP-22, ASP-7 ..., and the second thread wishes to pick an item, and ASP-22 is the first item that belongs to the second thread, then the ASP-22 is picked even though it is in position 3 in the queue. The only true queues are the ones in the collection that forms Q3.

The Examiner asserts that Hanif et al teach a plurality of stacks. In support of this assertion the Examiner cites col. 1, lines 58-67, which describe a prior art approach.

Basu 1-1

Strictly speaking, it is a valid showing of knowledge in the art. Applicants accept that the bare notion of stacks is known. However, one must be careful to not cavalierly combine this prior art teaching with whatever else Hanif et al propose for their system. Indeed, and stating more affirmatively, to the extent that the system proposed by Hanif et al does not use stacks, the fact that they are aware of this technique and chose not to use it, strongly indicates that it is not obvious to do so, or perhaps that it is inappropriate or not advantageous to do so.

Having established the existence of stacks in the prior art antedating the Hanif et al reference, the Examiner points to col. 4, lines 24-29 for the proposition that those prior art stacks include "a predefined set a predefined set of at least one mediation module that processes an applied signal to form a signal that is applied to said at least one resource of said collection of resources." It may be noted that this assertion relates to the first clause of applicants' claim 1.

The cited col. 4, lines 24-29 passage states:

> The requesting client 152 initiates a transaction by issuing a call to the ATP 120 and supplying the parameters of the request. Once the transaction request packet 158 is received by the socket 156 of the responding server 152, the transaction request is serviced and the responding server 152 returns a transaction response packet 160 reporting the transaction outcome

and a number of comments are appropriate relative to this passage.

First, it is noted that the above-quoted passage relates to sockets, as compared to some other elements. Furthermore, it relates to sockets that are involved in the proposed Hanif et al server. This paragraph does not relate to the specifics of implementing the the server – and particularly it does not relate to the AFPs, or the Q1, Q2, and Q3 collections of the Hanif et al FIG. 6.

Conclusion: The Examiner asserts that the Hanif et al sockets correspond to applicants' mediation modules.

Second, is clear that the col. 4 passage discusses the Hanif et al system, whereas the col. 1 passage that teaches stacks relates to some prior art system. The Examiner's coupling of the col. 1 teaching with the col. 4 teaching is logically not supported because there is nothing in the reference to suggest that the prior art stacks mentioned in col. 1 are

used anywhere in the Hanif et al system, and particularly in connection with the sockets that are addressed in col. 4.

Third, the col. 4 passage describes the establishment of one or more sockets between the sever and a client. The sockets *per se* do not even form a queue. They are merely a collection of objects. Certainly, the cites passage does not mention stacks, does not mention one or more modules that are stored in the stack, and of course does not mention that each stack has a predefined set of modules.

Conclusion: From the col. 1, or col. 4 citations, one cannot reach the conclusion that the sockets form stacks

The Examiner assertion that a socket corresponds to a mediation module is acceptable in the sense that claim 1 specifies different mediation modules and Hanif et al do have two types of sockets: the session listening socket (SLS) and the server session socket (SSS). Hanif et al show one SLS and a plurality of SSSs. It is also an acceptable assertion from the standpoint that sockets "processes an applied signal to form a signal that is applied to said at least one resource of said collection of resources," as claim 1 specifies. The SLS forms a signal that is applied to the module that creates SSSs, and an SSS creates a signal that is processed to fetch and transmit a file to the client. However, other considerations make the correspondence fail. The claim 1 stacks are particularly defined by:

> each including a predefined set of at least one mediation module that processes an applied signal to form a signal that is applied to said at least one resource of said collection of resources

This means that

1. each stack has one or more mediation modules,
2. each stack has a predefined set of those modules, and
3. these mediation modules process an applied signal that is applied to one or more resources (for example, hard drive, printer, a communication socket object, etc.) of the system that maintains the stacks.

The Examiner's assertion that the sockets are mediation modules meets only the third criterion. *They fail to meet the first two criteria.* For one, there is nothing about the collection of sockets that implies any ordering, or queue, so they cannot form elements of a stack. Furthermore, to establish a correspondence between the collection of sockets and

a plurality of stacks, one has to first divide the collection into groups (each group forming a stack). The Examiner has not done so. One could say that one group consists of the SSSs and another group consists of the SLS. That works for the SLS "stack" because the Examiner could assert that the single SLS, as a grouping by itself, forms a stack. The second grouping, however, is not a stack as specified by claim 1, not only because it is not a collection of objects without any ordering, but also because it is does not comprise a "predefined set" of mediation modules.

Conclusion: There are only two types of sockets, and so far into the analysis it is already established that the one set that has more than one member (SSS) clearly does not correspond to mediation modules, as specified in claim 1.

Over and above that, claim 1 specifies a "director" that intercepts requests that are directed to resources, classifies them, and directs them to different ones of the stacks. To employ the correspondence asserted by the Examiner, there must be something that serves as the "director," which captures requests to other resources, and directs them (instead) to one of the two alleged stacks that comprise the SLS and the SSSs. It is respectfully submitted that no requests come in that are first captured by something else and then are routed to the sockets. The sockets are the objects that receive the requests in the first instance.

Conclusion: Since sockets are the objects that receive input signals rather something receiving signals and directing them to the sockets, the sockets cannot correspond to the mediation modules.

To the extent that the Johansson reference is not directed to a teaching of a plurality of stacks, which the Examiner is not asserting that it does, the above constitutes another reason to hold that claim 1 is not obvious in view of the Hanif et al and Johansson et al references.

The Examiner offers the Johansson et al reference for the proposition that it teaches "that each different one of said resources are responsive to requests of a different type," quoting from the reference the following teaching:

> another aspect of the invention relates to efficient allocation of resources from different pools of resource units. A request for resources very often involves allocation of different types of resource units.

Basu 1-1

Apparently, the Examiner makes this reference in an effort to show that Johansson et al suggest the service director of the second clause of claim 1.

First, the fact that there are pools of resource units, and that Johansson et al efficiently allocate them does not teach that different ones of the resources are responsive to requests of a different type. Second, even if the Johansson et al reference teaches that which the Examiner asserts, it falls woefully short of the second clause of claim 1, which specifies a service director module that does the following:

1. intercepts requests of different types that are directed to the resources,
2. classifies said requests in accordance with said types of said requests, and
3. directs said requests to different ones of said processing stacks, based on said classifying

Relative to claim 1, the Examiner has not pointed to any element in Hanif et al that allegedly operates as the "service director." Further, there is no element in Hanif et al that intercepts requests of different types that are directed to the resources, classifies them and directs them to alleged stacks that contain the sockets as the mediation modules.

The Johansson et al reference also does not have such an element, and the Examiner has not pointed to any. Moreover, even if Johansson had such an element, there is no motivation for adding such an element to the Hanif et al arrangement. The latter neatly handles the session opening requests by establishing SSSs, and handles all transaction requests by creating the AFP objects and satisfying the requests as described above. There is no need for, or any benefit in, creating a service director that classifies requests and directs them to different socket groupings, whether they are stacks or not. **Conclusion: Since the combination of Hanif et al and Johansson et al do not teach or suggest a plurality of stacks like the ones defined in claim 1, and do not teach or suggest the service director as defined in claim 1, it is respectfully submitted that claim 1 is not obvious in view of these references.**

Claim 2-6, 12-15, and 25 depend on claim 1 and, therefore, theses claims are also not obvious in view of these references.

Regarding claim 2, the Examiner's assertion that the limitation of claim 2 is met because a socket receives a signal from some resource in the server (e.g., a file to be communicated) and that signal is accepted by the socket, would be valid, **but for the fact**

that, for the reasons expressed above, a socket is not a mediation module in the sense defined in claim 1. Therefore, claim 2 is no more rejectionable in view or the cited art than claim 1 is. Stated affirmatively, claim 2 is believed to be not obvious in view of the cited references.

As for claim 3, it specifies that at least one processing stack comprises an ordered sequence of at least two mediation modules. The Examiner points to col. 5, lines 21-34, but those lines discuss the AFP objects (AFP sessions and the AFP requests), and not the sockets. In other words, it appears that the Examiner is "changing horses in midstream." **Respectfully, the Examiner cannot in connection with claim 1 that assert the *sockets* correspond to the mediation modules, and then in connection with claim 3 assert that the *AFP objects* correspond to the mediation modules.**

Assuming that the Examiner reverses himself on the claim 1 assertion, and now asserts that the AFP objects are the mediation modules, since the AFP objects are directed to Q3, this is tantamount to an assertion that Q3 forms the stacks. Applicants respectfully submit that Q3 does not contain stacks as specified in claim 1, because although eQ3 contains a plurality of queues, each queue does not have a predefined set of at least one mediation module. Rather, each queue has a number of AFP objects (mediation modules) that varies with the respective session's whim.

Conclusion: Asserting that the AFP objects form the mediation modules of claim 1 does not comport with the claim 1 specification and, therefore, claim 3 is not obvious when the "sockets" = "mediation modules" correspondence is asserted (as in connection with claim 1) and is also not obvious when the "sockets" = "AFP objects" correspondence is asserted (as in connection with claim 3).

As for claim 4, which specifies that the service director receives a request from an application that is active on the system, the Examiner cites col. 5, lines 12-34. The cited passage teaches that a multi-threaded process has multiples scheduable threads of control, and that the threads are time-sliced on the processor. That does not teach that teach anything about a service director, and even if one were to assume that the processor, or the multi-threaded process, is the "service director," it still remains that there is no teaching of processing "a request from an application that is active on said system." Therefore, the assertion by the Examiner relative to claim 4 fails.

Basu 1-1

As for claims 5 and 6, they depends on claim 4 and since claim 4 is not suggested by, and is not obvious in view of the cited references, it follows that claims 5 and 6 are also not suggested by, and are not obvious in view of the cited references.

As for claim 12, the Examiner asserts that the passage at col. 4, lines 30-56 teaches a service request classifier. Applicants respectfully disagree. *The passage merely reports that there are two types of sockets for receiving two different types of requests.* There may be an element in the Hanif et al system that makes the determination as to which type of request was presented (a request to establish a session, or a transaction request), but the cited passage does not reveal what that element is, and the Examiner does not identify what that element is (if there is such an element).

Regarding claim 13, which specifies that the classification is made not only based on the type of request but also on arguments of the service request, the Examiner points to the passages at col. 4, lines 24-29, which teaches that the client initiates a transaction by making the appropriate request and supplying parameter values. Supplying parameter values so that the service provided would be correct does not mean that a classification process is carried out by considering the parameter values. There is no indication in Hanif et al that those parameters are used to classify requests. Indeed, the only distinction in connection with requests is to which session they belong, and that does not correspond to transaction parameters. It may be noted that Hanif et al do not describe a variety of service request types. All requests are embodied in AFP modules. Hence, there is no classifying based on type of request, and certainly there is no classifying based on arguments of the service requests.

The best that can be surmised is when a request comes in, it is identified as to whether it is a request for a known session, in which case it is directed to one of the Q3 queues, or it is a request from an unknown session, in which it is directed to a module that creates a new SSS, and an new queue in Q3. This is not explicitly taught in Hanif et al, but only a surmise. It may be in error but, then again, Hanif et al do not describe any process of classifying and directing of requests. As for the directing of requests to Q3 queues, the above remarks demonstrate that the various Q3 queues cannot constitute the stacks of claim 1. The above remarks also demonstrate that the SLS and SSSs also cannot constitute the stacks of claim 1.

Basu 1-1

Claims 7-8, 10-11, and 23-24 were rejected under 35 USC 103 as being unpatentable over Hanif et al in view of Johansson et al, and further in view of Hershey et al, US Patent 5,414,833. Applicants respectfully traverse.

Claim 7 specifies that at least one mediation module (in one on of the stacks) is based on a chosen security policy. The Examiner quotes a portion of a passage from col. 23, lines 33-65 of the Hershey et al reference, and follows it with the phrase "wherein it is discussed above that an application or software module could be construed as a mediation module."

Applicants are at a loss.

First, it does not appear to applicants that anywhere earlier than item 7 of the Examiner's Detailed action did the Examiner construe, or assert a correspondence to, some "application or software module" being the mediation module. Unless, of course, the Examiner is referring to the *socket objects*, regarding which the Examiner made assertions relative to claims 1 and 2, or the *AFP objects*, regarding which the Examiner made assertions relative to claim 3. Without knowing precisely what software elements the Examiner is referring to, no cogent response can be offered.

Second, Hershey et al teach in the cited col. 23 passage that the described arrangement includes a security application that assumes that encryption is performed. The data portions of transmitted messages are monitored to ensure that plaintext is not transmitted in situations where ciphertext is expected. The difficulty that this passage notes is that plaintext is hard to detect when data compression is executed. The passage concludes that the problem is not insurmountable because compressed data nevertheless typically includes some plaintext portions.

Applicants respectfully submit that this teaching is simply not relevant. It does not suggest that either the socket objects, or the AFP objects ought to be encrypted, or checked to make sure that they are encrypted. Moreover, encrypting these objects (or making sure that they are encrypted) makes no sense. For example, if a client wishes to communicate securely, and if the server agrees to accept an encrypted transaction request, it would make no sense to create an encrypted AFP object. Rather, it would be much better to allow encrypted communication, decrypt the communication once it is received,

and create an AFP object in plaintext. There is simply no reason to encrypt sockets, or to encrypt AFP objects.

The Examiner points to various problems that may b associated with breaches, but the server described by Hanif et al merely accepts requests, constructs AFP objects from the requests, and services the requests. Since the AFPs are constructed locally within the server, it is quite simple to insure that no AFP is constructed that could cause damage (i.e., to limit the type of request that is honored). As indicated above, the most that may be desired is to handle communication requests that are encrypted. As far as this goes, all that is needed is a decryption module, and there is particular reason to implement a decryption module within a socket object (if the Examiner maintains the "sockets=mediation modules" assertion), or within an AFP (if the Examiner maintains the "AFPs=mediation modules" assertion).

For these reasons, applicants respectfully submit that claim 7 is not obvious in view of the Hanif et al, Johansson et al, and Hershey et al combination of references.

As for claim 8, it specifies that the mediation module performs encryption; i.e., converts plaintext to ciphertext. There is absolutely no need for a socket of the server, which receives requests, to perform encryption. There is also absolutely no suggestion to the effect that an AFP object, which is created in response to a request, would perform encryption. Applicants respectfully submit, therefore, that claim 8 is not obvious in view of the Hanif et al, Johansson et al, and Hershey et al combination of references.

Regarding claim 10, there is no suggestion anywhere in Hanif et al that authentication would be advantageous, or desirable; and the fact that the notion of authentication is present in the art is not a sufficient motivation for doing so. (Essentially all inventions are formed from known notions, but their combinations are novel.) The notion of authentication has been known for many decades before the creation of the Hanif et al system and yet, Hanif et al did not deem it meritorious enough for even an off-hand comment. As for the different "mediation modules" that the Examiner asserts in connection with claims 1 and 3, i.e., the socket objects, and the AFP objects, it would not make sense to have the AFP objects perform the authentication, even if authentication were desired. Rather, it would make much more sense to have a separate software module perform this function. Applicants respectfully submit, therefore, that claim 10 is

not obvious in view of the Hanif et al, Johansson et al, and Hershey et al combination of references.

Regarding claim 11, none of the elements that the Examiner suggests as corresponding to the "mediation modules" is, or can be, a file system. The "Official Notice" assertion that it is well known to implement a secure file system is wholly irrelevant.

In the remarks pertaining to claim 11 the Examiner asserts "a mediation module is in essence any software module that acts as a go between for a client and system resource." Effectively, this is an assertion that all, and each, of the software modules in a server are "mediation modules," because in *one sense or another* all software modules serve as a go-between between a client and system resources. This is way too expansive a definition for a mediation module. While it may be logically acceptable in the abstract, it certainly is not acceptable in the context of applicants' specification, and in the context of applicants' claims where the mediation objects are in stacks, and other limitations pertain to the mediation modules. Clearly, for example, though the service director is a software module, and it serves as a go-between clients and resources of the server, it is not a mediation module. In any event, whatever software modules the Examiner wishes to assert as corresponding to the mediation modules is permitted, but that does not relieve the Examiner from *clearly establishing his position* as to what elements constitute the mediation modules.

Request: Because applicants are entitled to a consistent, clear, and fixed assertion of correspondence to which an argument can be directed (i.e., a clear and fixed target), the Examiner is respectfully requested to clearly elucidate what allegedly corresponds to the mediation modules, and what allegedly corresponds to the different stacks.

As for claim 23, there is simply no hint of a suggestion in Hanif et al that it might be beneficial for mediation module to include an authentication code retriever. The Examiner cites col. 17, lines 8-56 of the Hershey at al reference, and quotes that the a security alert message transmission means causes a message authentication code to be transmitted. However, that does not provide the suggestion, or the motivation, to create a mediation module (whether it is the socket object, or the AFP object) to include anything

Basu 1-1

that constitutes an authentication code retriever. Applicants respectfully submit, therefore, that claim 23 is not obvious in view of the Hanif et al, Johansson et al, and Hershey et al combination of references.

As for claim 24, no hint of an **administrative user** is found in any of the references, and the Examiner has not pointed to any. The Examiner recites the benefits that a sandbox and the notion associated with all service requests going through the direction, but *those are the attributes of applicants' invention*. They are not attributes that are described or suggested in any of the references cited in connection with claim 24. Applicants respectfully submit, therefore, that claim 24 is not obvious in view of the Hanif et al, Johansson et al, and Hershey et al combination of references.

Claim 9 was rejected under 35 USC 103 as being unpatentable over Hanif et al in view of Johansson et al, and further in view of Traversat et al, US Patent 6,052,720. Applicants respectfully traverse.

Claim 9 specifies that the mediation module is a name space manager. The Examiner asserts that Traversat teaches the notion of a mediation module being a namespace manager, citing col. 7, lines 31-36. The cited lines state:

> Each namespace is managed by a default namespace manager. The namespace manager control how the entries are stored and accessed within the namespace. The manager implements a standard interface that exports the security, storage, and ownership attributes of any entry in the namespace.

Respectfully, this passage does not suggest that any element of the Hanif et al elements that allegedly correspond to the "mediation modules" may be, ought to be, or could somehow benefit the arrangement if it were, a namespace manager. Even if one were to take the inappropriately expansive notion of the Examiner that any software modules is a mediation module, it still remains that nothing in Traversat et al suggests that a mediation module of the type specified in applicants' claim 1 (e.g. that is placed in a stack) can be a namespace manager. Furthermore, it makes absolutely no sense for a socket object, or for an AFP object to be a namespace manager. Therefore, it is respectfully submitted that claim 9 is not obvious in view of the Hanif et al, Johansson et al, and Traversat et al combination of references.

Basu 1-1

As for claims 16-18, applicants respectfully submit that these claims are not obvious in view of the cited references at least by virtue of the fact that they depend on claim 1.

Claims 19-22 were rejected under 35 USC 103 as being unpatentable over Hanif et al in view of Johansson, and further in view of Bond et al, US Patent 6,275,938. Applicants respectfully traverse.

Applicants respectfully repeat the arguments presented in the last Office action, and also respectfully disagree with the Examiner remarks in item 18. While it is true that WHKRNL32 is placed outside the sandbox so that a rouge applet would not be able to compromise security, it does not follow that WHKRNL32 is adapted to receive security information from outside the system. The reason that the Examiner's conclusion does not follow is because the WHKRNL32 could have a static set of instructions and parameters, or it could have a set of instructions and/or parameters that are altered pursuant to a policy contained within the system. There is nothing that requires, or suggests that WHKRNL32 does, or ought to, receive policy information from the outside. Moreover, the notion that it is important for WHKRNL32 to be outside the sandbox so that a rouge applet would not be able to compromise security *does not controvert* applicants' observation that WHKRNL32 implements the security policy and does not appear to be modifiable at all. Perhaps it could be, but it is not so described, or suggested.

Claims 26-29 were rejected under 35 USC 103 as being unpatentable over Hanif et al in view of Johansson et al, and further in view of Eliott. Applicants traverse, and respectfully direct the Examiner's attention to the earlier remarks pertaining to the Eliott reference.

Additionally, applicants respectfully submit that there is no motivation for adopting the teachings of Eliott to the Hanif et al system. The Eliott system is directed to electronic games that are purchased, typically by, or for use by, young people. The threat of game piracy is real, and the Eliott effort is to prevent game playing except with authenticated games. To that end, the expansion device was introduced that enhances the security. The Hanif et al system, on the other hand, is a server that is under control of a

party and it accepts requests only from the network. It is a totally different milieu. Not only is the risk of receiving bogus, damaging requests low, it is extremely easy to guard against them (relative to the protection of games that are executed on console that are not under physical control of the provider). The risk that the operating system of the server is compromised is orders of magnitude lower than with games consoles/personal computers that are under physical control of the potential wrong-doer. Applicants submit, therefore, that there is no motivation for adding an ROM to Hanif et al that contains an operating system that is read upon boot-up and authenticates programs that are contained on the server. Therefore, it is respectfully submitted that claims 26-29 are not obvious in view of the Hanif et al, Johansson et al, and Eliott combination of references.

Claims 32-34 were rejected under 35 USC 103 as being unpatentable over Eliott in view of Hanif et al and in view of Johansson et al. Applicants respectfully traverse.

Aside from the fact that claim 30 is not anticipated by Eliott, it is also respectfully submitted that Hanif et al combined with Johansson et al do not teach a reverse sandbox, and therefore, the combination of Eliott, Hanif et al and Johansson et al do not create a storage medium that includes instructions for verifying software, and certainly do not create a storage medium that not only verifies software, but that software implements a reverse sandbox in the system. Therefore, applicants respectfully submit that claim 32 in not obvious in view of the Eliott, Hanif et al and Johansson et al combination of references. The same argument applied to claim 33, and claim 34. It simply makes no sense to introduce into the Eliot game the notions of sockets, and AFP objects. There are no statements in Eliott that might suggest that, and there are no statements in either Hanif et al or Johansson et al that might suggest it. Therefore, applicants respectfully submit that claims 32-34 are not obvious in view of Eliott, Hanif et al, and Johansson et al combination of references.

Regarding the Examiner's "Response to Arguments," applicants respectfully disagree that Hanif et al combined with Johansson et al disclose processing stacks that have a predefined set of at least one mediation module. Applicants also note that while it is acceptable to view a request for establishing a session and a request to execute a

transaction as different types of requests, one should be keep in mind that even if it could be argued that the transaction requests are placed in stacks (which applicants dispute that the **fully meet all** of the claim 1 limitations), the transaction requests are but **one type of request.** The other type of request, that of establishing a session are clearly not executed with the aid of any stack, or even a queue. Except for the above, Hanif et al do **not** disclose receiving different types of requests (as asserted by the Examiner at top of page 20). Also, the statement at page 20, lines 5-6 that Hanif et al disclose " a multithreaded system, where each thread has its own processing stack" is simply not correct. While it is not clear what the Examiner considers to be the stacks, it is perfectly clear the only elements that the Examiner explicitly mentioned in a manner that suggests that they are stacks, which is Q1, Q2, and Q3, DO NOT correspond in number to the number of threads. Lastly in connection with item 14 of the Examiner's remarks, applicants respectfully disagree that motivation exists "not only to protect resources from intruders by preventing access to the stacks, but also allows the designer of the system more flexibility in that more specialized thread method can be defined that are only operable for one type of resource." Respectfully, this applies too much hindsight, after having read applicants' application.

It is helpful to remember that other than a single mention of stacks in connection with prior art systems, Hanif et al do not even mention stacks. The multi-threaded operation is not created in order to have "specialized threads." They are strictly created to speed up operation and to provide fairness in the provision of service. The work performed by each thread is the same as the work performed by the other threads. This is quite evidently so, since any active session can be assigned to any thread. Moreover, flexibility is simply not one of the issues that arises in an arrangement that is designed to service client requests, and does so with a multi-threaded process. It is a limited-purpose arrangement, and there is nothing to suggest that additional "flexibility" is called for.

As for the comments in item 15, applicants respectfully submit that the test to be applied is NOT whether something could be done in the manner that it is specified in applicants' claims, but whether there is anything in the base reference that is suggestive of the desire not only to do something, but to do it in the manner defined in applicants' claims. Thus, for example, the Examiner states the "it is felt that such a security policy

could be defined within the processing stack...." Perhaps so. Certainly, in the context of applicants' disclosure, that was done and, obvious, therefore, it "could be" done. The issue is whether there is anything to suggest that it should be done, and further, whether there is anything to suggest that it should be done in this manner. Respectfully, as explained above, the Hanif et al system, to the extent that the socket objects, or the AFP objects, constitute the mediation modules, a security policy either actually cannot be done "within the processing stack" or it would have been more advantageous to do it elsewhere.

In conclusion, applicants respectfully submit that a number of inconsistencies, or at least ambiguities, exist in the Examiner's rejection, which that have hampered the preparation of this response. If the Examiner remains unconvinced that the outstanding claims are allowable, applicants respectfully request that the Examiner clearly identify the elements that the Examiner considered to be the "mediation modules," the structures that form the "plurality of stacks" point to the attributes that make those elements "stacks," and identify the "service director."

In light of the above amendments and remarks, applicants respectfully submit that all of the Examiner's rejections have been overcome. Reconsideration and allowance are respectfully solicited.

Respectfully,
Anindya Basu
Suvo Mittra

Dated: 2/8/04

By _____
Henry T. Brendzel
Reg. No. 26,844
Phone (973) 467-2025
Fax (973) 467-6589
email brendzel@comcast.net